Time series prediction evolving Voronoi regions

Cristobal Luque · Jose M. Valls · Pedro Isasi

Abstract Time series prediction is a complex problem that consists of forecasting the future behavior of a set of data with the only information of the previous data. The main problem is the fact that most of the time series that represent real phenomena include local behaviors that cannot be modelled by global approaches. This work presents a new procedure able to find predictable local behaviors, and thus, attaining a better level of total prediction. This new method is based on a division of the input space into Voronoi regions by means of Evolution Strategies. Our method has been tested using different time series domains. One of them that represents the water demand in a water tank, through a long period of time. The other two domains are well known examples of chaotic time series (Mackey-Glass) and natural phenomenon time series (Sunspot). Results prove that, in most of cases, the proposed algorithm obtain better results than other algorithms commonly used.

Keywords Time series · Artificial intelligence · Evolutive algorithms · Evolution strategies · Machine learning · Voronoi regions

C. Luque (✉) · J.M. Valls · P. Isasi
Dept. de Informatica, Universidad Carlos III de Madrid,
Av. Universidad 30, 28911 Madrid, Spain
e-mail: cluque@inf.uc3m.es

J.M. Valls
e-mail: jvalls@inf.uc3m.es

P. Isasi
e-mail: isasi@ia.uc3m.es

1 Introduction

Many artificial and physical phenomena can be modelled with time series. This fact makes time series prediction problem as complex as interesting. There are many methods able to tackle these problems, but in most cases these methods only look for a general approach for the series behavior. The main problem is that time series generally involve local behaviors that do not allow a good level of prediction using a global approach. This paper presents a method for finding local behaviors which can make predictions at these levels, and thus, achieve a better total prediction. The method presented is based on the division of the input space into Voronoi regions using Evolution Strategies.

In the field of time series forecasting, ARIMA [4] and regression [19] are the most used tools. But those linear systems have a reduced capability of adaptation, mainly on highly non-linear problems. This fact has motivated researches about other kind of tools based on Artifical Intelligence for Time Series forecasting, as Neural Networks [20] and Expert Systems [23].

Artificial Neural Networks have been proved to be effective methods for predicting time series [21, 29]. In [29], we can find a time series analysis using nonlinear dynamic systems theory and models based on multilayer feed-forward networks. These models are applied to data measures of the tide level in the Venice lagoon over the years 1980–1994. In recent works [8, 26], learning methods are used to automatically select the most appropriate patterns for training, depending on the example to predict. This training method uses a lazy learning strategy that builds local approximations centered in the new patterns. Galvan et al. [8] apply their method to the Mackey-Glass and Venice lagoon time series. Further works have applied Evolutionary Algorithms

[6, 11], as gene expression programming [30] and cardinal B-Spline models [27] to the Time Series prediction area.

There are several algorithms that divide the input space in different regions. LVQ [25] uses Voronoi regions for classification tasks. Radial Basis Neural Networks (RBNN) [17], in one of the classical non-supervised methods to determine the centers of the Radial Basis functions, divide the input space in Voronoi regions too, using K-means [10, 14] or other similar algorithms.

RBNN also use alternative ways to determine the centers of the Radial Basis functions, adjusting their positions in a supervised manner in order to minimize the output squared error. This idea is closer to the one presented in this paper: moving the regions while error minimization process. The use of genetic algorithms evolving Voronoi Regions for classification problems is also presented in [5]. Packard used genetic algorithms [9] to face the problem of predicting dynamic systems: in [15, 16, 18], a new approach is suggested, selecting subsets of the input space by means of conditional rules for time series prediction. We used Packard's idea to improve his approach in previous works [12, 22].

2 Objectives

In this paper, we present a new supervised learning system based on Evolution Strategies [1, 2, 24] in order to perform prediction tasks. Usually, this kind of systems rely on the whole training data set to derive the procedure used to make predictions. However, in some domains, the peculiarities of the input space favor approaches that stress the importance of local information. Among these, we could mention the existence of significant differences between different regions of the input space.

The system introduced in this work tries to automatically detect the areas in the input space that share enough features to be accurately predicted by the same method. In addition to that, the system will create a linear predictive model appropriate for each of them. Thus, at the end, it is going to produce, on one hand, a partition of the input space in different areas by means of a set of prototypes and the nearest neighbor rule, and, on the other hand, a linear model to perform the predictions corresponding to that region.

As in most machine learning algorithms, two stages are required: a training stage to create a model, and a test stage, to validate the model. As part of the training stage, the input space is divided in Voronoi regions by means of a set of prototypes randomly initialized. In each of these Voronoi regions, a linear regression is performed to fit the points representing the instances of the training set, defined by pairs (input, output). Besides, a minimum number of points will be required for each regression, otherwise it will be considered not reliable and will not be allowed to produce a prediction. Thus, several local linear regressions are performed, one per region. These regions will be iteratively adjusted in order to make the predictions of the regressions associated to each region as accurate as possible. This adjustment will be done using Evolution Strategies.

The algorithm has been conceived in a way that tends to allocate noisy patterns into specific regions, hence improving the accuracy of the models fitted to the patterns that show a clearer structure. Therefore, a predictive model composed of several adjusted regions is built. In each of these regions a linear regression is performed. As we explain in the following section and with the aim of improving the prediction quality of the system, instead of using only one model, our method will build several models (we call them subsystems) with different random initializations. In this way, the confidence and accuracy of the prediction will be significantly improved.

When a test pattern needs to be predicted, it will be assigned to the appropriate region of each subsystem, following the nearest neighborhood rule. Then, each subsystem will provide a prediction based on the local regression associated to that region. In certain subsystems, the corresponding region might not be able to make a prediction because it does not have the required amount of training points. This is not a problem because other subsystems could make the prediction. The final output of the system will be the average of the valid predictions.

In order to store local information in the individuals, a Michigan approach [3] has been implemented in the Evolutionary Algorithm, using a Steady-State strategy. In the Michigan approach, the solution to the problem is the total population instead of the most fitted individual. This way allows individuals parallel evolving, focusing on local data.

The rest of the paper is organized as follows in Sect. 3 the proposed method is described in detail. The experimental validation of the method is presented in Sect. 4 and, finally, the conclusions of the work are explained in Sect. 5.

3 The method

The training process divides the input variables space into Voronoi regions. Let n be the dimension of the input space, and thus $n+1$ the dimension of the pattern space, being the extra dimension the prediction. We will call "prototype" to a point P (a vector) in the input variables space (a subset of, R^n). Given a set of k prototypes $\{P_i \in R^n$, where $0 < i \leq k\}$, this set divides R^n in k Voronoi regions, when the nearest neighborhood rule is used. This means that a point belongs to the region defined by a prototype if this point is closer to this prototype than any other prototype. If we call V_i to the region defined by P_i, we can describe this region mathematically as in (1).

$$V_i := \{x \in R^n / d(x, P_i) < d(x, P_j) \text{ for all } j \neq i\} \quad (1)$$

Here d is the standard euclidean distance. The goal of this stage is to determine the best partitioning in terms of predictive accuracy. The approach suggested to do this is based on an evolutionary process. Each prototype is represented by an individual in an Evolution Strategies system. The learning procedure of the Evolution Strategies moves those prototypes to place them in the location where the predictive properties are optimized. The predictive properties of the regions are used as the fitness value of the individuals.

3.1 Encoding

In order to evolve individuals, we have chosen Evolution Strategies, because this evolutionary approach is the best suited to work with real values variables. For this problem, the real values to adjust are the coordinates of the prototype represented by the individual. In Evolution Strategies, individuals need an additional parameter, representing the variance on the mutation. That variance means how close from the solution those individuals are: a small variance means that the individual is very close to a solution of the problem; a big variance means that the individual is far from a good solution, and thus it needs big changes.

Our final chromosome schemata for an individual I could be expressed as:

$$I = (x_1, \ldots, x_n, \sigma_I)$$

where $x_1, \ldots, x_n$ are the coordinates of the prototype this individual represents (we name it P_I), and σ_I is the variance for this individual.

As we explained previously, for each individual I, the prototype P_I defines a Voronoi region V_I (1). After that, a linear regression R_I is calculated with the training patterns belonging to V_I.

3.2 Fitness evaluation

Training patterns are composed of vectors of $(n+1)$ dimensions (n dimensions for the input variables and one dimension for the output). For the encoding of the individuals, just the n input dimensions are used. Let $T = (t_1, \ldots, t_{n-1}, t_n, t_{n+1})$ be a training pattern. A projection map from R^{n+1} to R^n is defined as in (2):

$$\Pi(t_1, \ldots, t_{n-1}, t_n, t_{n+1}) := (t_1, \ldots, t_{n-1}, t_n) \tag{2}$$

In order to compute the fitness value of an individual (prototype P_i), first we need to assign a set of pattern projections to that prototype, following the nearest neighborhood rule. It can be seen in (3) for a given pattern T:

$$\Pi(T) \in V_i \iff d(T, P_i) < d(T, P_j) \quad \text{for all } j \neq i \tag{3}$$

Once every training pattern is assigned to its corresponding region, a regression R_i is calculated for each region V_i. Thus, R_i is a linear regression of the output variable over the input variables for each pattern T, such that $\Pi(T) \in V_i$. Let $R_i(T)$ be the estimated output for the pattern T by the regression R_i, and Out the output value of the pattern T. That is, if $T = (t_1, \ldots, t_{n-1}, t_n, t_{n+1})$, then $Out(T) := t_{n+1}$. Then, the error of that estimation is $E_T = |Out(T) - R_i(T)|$. Thus, we have the relationship $P_i \rightarrow V_i \rightarrow R_i$ for each $i \leq k$. That is, a prototype P_i defines a region V_i, and that region has an associated regression R_i. All this process is represented in Fig. 1.

The goal of the algorithm is to minimize the sum of errors. In standard Evolutionary Algorithms, a fitness value is assigned to each individual. For instance, we could use the sum of the errors for all the patterns associated to the region V_i as the fitness function for each individual (prototype P_i). However, we consider that it would not be suitable in our approach since the individuals with fewer associated patterns (that is, fewer projections in their region, and therefore, a sum of less error terms) would tend to behave better. Another candidate for fitness function could be the previous one divided by the number of patterns belonging to the region associated to the individual (mean error). This choice would not be much better that the previous one, because it would allow individuals with more patterns in their regions to evolve stealing patterns from the nearby regions, and thus, the error derived from using a regression that is not suitable would be compensated by the fact that the magnitude of such error would be shared among all the data points in the region. The dilution of the consequences of the mentioned undesired behavior, has led us to look for another alternative, a population-based fitness. That means that instead of having a fitness value for each individual, we have a fitness value for the whole population. This value is the sum of the errors $E_T = |Out(T) - R_i(T)|$ for each pattern T and i such that $\Pi(T) \in V_i$. Thus, mathematically, it can be written as in (4).

$$\text{Fitness} := \sum_{T} |Out(T) - R_i(T)|, \quad i \text{ such that } \Pi(T) \in V_i \tag{4}$$

3.3 Evolution

The initial population is randomly created. The evolution process basically works as a $(1+1)$ parallel Evolution Strategy. That is, in every generation, each parent produces an offspring by mutation. Let $I = (x_1, \ldots, x_n, \sigma_I)$ be an individual and let $I' = (x'_1, \ldots, x'_n, \sigma'_{I'})$ be its offspring. The mutation process can be expressed as in (5) and (6):

$$x'_i = N(x_i, \sigma_I) \tag{5}$$

$$\sigma'_{I'} = \sigma_I e^{\alpha N(0,1)} \tag{6}$$

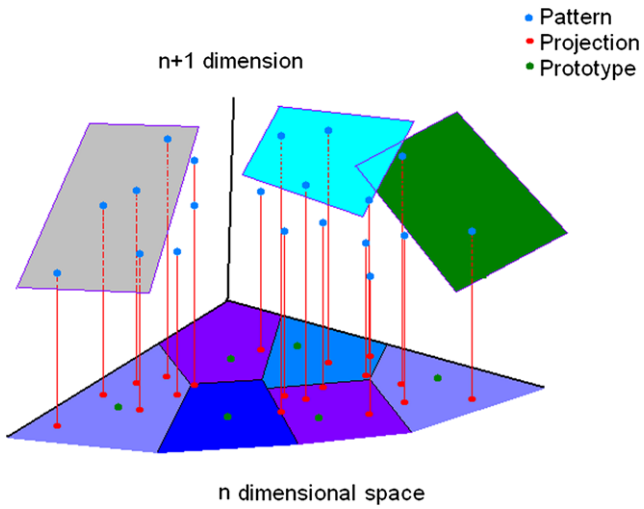

Fig. 1 Graphical representation of the Algorithm

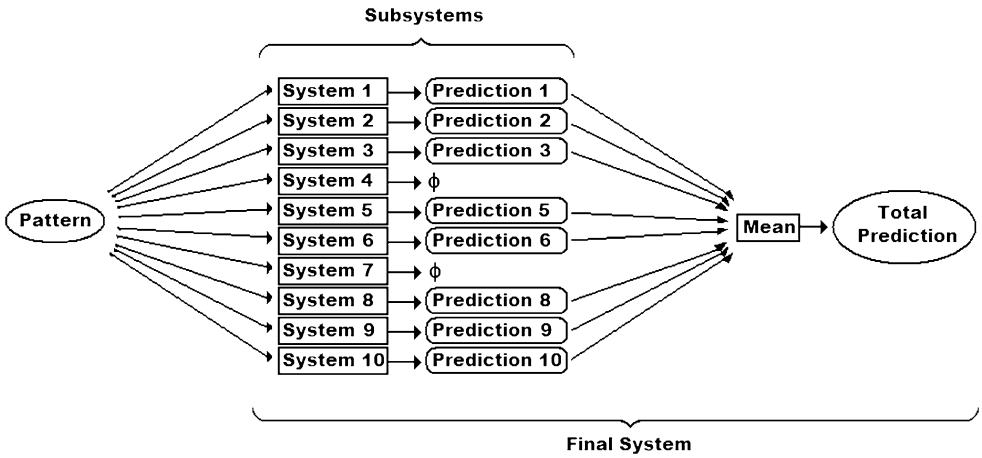

Fig. 2 Schema of the two phases of prediction of the Evolutive System

Table 1 Configuration for the experiments for the Water Depot TS

Domain	Water Depot
Training set	1500
Validation set	499
Normalization	[0,1]
Input variables	8
Prediction horizon	1

where $N(X, Y)$ represents a normal random variable with mean X and variance Y, and $\alpha = 0.7$ is a constant. The variables σ_I and $\sigma'_{I'}$ are the variance of the parent and offspring respectively. Those equations are extracted from [1].

For each offspring I' we look for the closest individual $\hat{I}$ in the population in terms of euclidean distance. Both individuals must be compared in order to select the best, so we calculate the population fitness before and after replacing $\hat{I}$ by I'. Finally, if the population fitness is worse after the replacement, we undo the change and keep the population as it was before. The whole training process is schematically described in Algorithm 1.[1]

3.4 Building the system and prediction of testing patterns

This evolutionary approach allows us to build a predictive model composed of several local linear regressions associated to the specific regions.

With the aim of improving the quality of the predictions, instead of one, ten models or subsystems with different random initializations will be built, in order to make the predictions more reliable and accurate. The reason behind this is that a regression is able to produce a reliable prediction only if the number of points it has been built with is big enough. Because of this, the regions composed of a scarce number of points will not be allowed to produce an output.

[1] For an optimized version of the algorithm, not all the Regressions need to be recalculated in each fitness evaluation. Regressions have a high computational cost, so part of this operations can be stored in order to make the algorithm faster. This version of the algorithm is a simplified one, thus it does not use optimization.

Table 2 Error measures of the experiments with the EVP for the number of regions for the Water Depot TS

Reg.	Threshold 3		Threshold 5		Threshold 10	
	MSE	% pred.	MSE	% pred.	MSE	% pred.
3	0.003142	100.00	0.003142	100.00	**0.003137**	**100.00**
6	0.003240	100.00	0.003229	100.00	0.003240	100.00
8	0.003346	100.00	0.003337	100.00	0.003313	100.00
10	0.003442	100.00	0.003356	100.00	0.003337	99.96
12	0.003477	100.00	0.003428	100.00	0.003413	99.94
16	0.003582	100.00	0.003443	99.70	0.003443	98.70
20	0.003803	98.96	0.003588	97.17	0.003411	92.81

Algorithm 1 Training Algorithm

variables
Set of individuals: $\{I_1, \ldots, I_m\}$
Set of offsprings: $\{I'_1, \ldots, I'_m\}$
Set of regions: $\{V_1, \ldots, V_m\}$
Set of regressions: $\{R_1, \ldots, R_m\}$
Set of training patterns: $\{T_1, \ldots, T_s\}$
Number of generations: $g = 0$
RandomInicialization $\{I_1, \ldots, I_m\}$
while $(g < GENERATIONS)$
  for $(i = 0)$ **to** m **do**
    $I'_i = Mutation(I_i)$
    $I_i = SelectBest(i, I_i, I'_i)$
  end for
end while

---

procedure SelectBest(integer i, individual J, individual J')
  $Set\,J = \{I_1, \ldots, I_{i-1}, J, I_{i+1}, \ldots, I_m\}$
  $Set\,J' = \{I_1, \ldots, I_{i-1}, J', I_{i+1}, \ldots, I_m\}$
  if $fitness(set\,J) < fitness(setJ')$ **then** $SelectBest = J$
  else $SelectBest = J'$
end procedure

---

procedure fitness(Set of individual $\{J_1, \ldots, J_m\}$)
  $fitness = 0$
  CreateRegions$(V_1, \ldots, V_m)$ with $\{J_1, \ldots, J_m\}$
  CalculateRegression$(R_1, \ldots, R_m)$ with
    $\{V_1, \ldots, V_m\}$ and $\{T_1, \ldots, T_s\}$
  for $(k = 0)$ **to** s **do**
    find j **such** $\Pi(T_k) \in V_j$
    $fitness = fitness + |Out(T_k) - R_j(T_k)|$
  end for
end procedure

Therefore, our system is composed of ten subsystems, as it can be seen in Fig. 2. Each one is a prediction model randomly initialized and evolved with the training patterns, as it has been explained above. Some subsystems might not produce a valid output because the corresponding Voronoi regions are not reliable enough. However, the final output of the whole system is the mean of the valid—and reliable—outputs of the other subsystems. The whole system is now likely to produce reliable predictions for new testing patterns.

Algorithm 2 Prediction Algorithm

STEP 1: find i such that $\Pi(T) \in V_i$
STEP 2: if $(\sharp Ri >$MIN$)$ **then**
    OUTPUT:$= R_i(T)$
  else NO-OUTPUT

Let us summarize the prediction process: For a given testing pattern T, each subsystem must establish, in the first place, the region in which the pattern is located. Then, the prediction is calculated as the estimated value of T by the regression that corresponds to that region.

The prediction process for each subsystem could be summarized as in Algorithm 2, where $\sharp R_i$ is the number of points in the region R_i after the training process and *MIN* is the minimum of points required for a prediction. The *MIN* parameter depends on the accuracy we desire for the algorithm. For linear regressions, in order to make its prediction reliable, it is usually recommended to be calculated with, at least, 5 points for each variable. Therefore, the number of input variables multiplied by 5 is a good value for *MIN*.

For a given pattern, each subsystem may produce an output or not. The final output of the system for a pattern T is the mean of the valid values returned by each subsystem, as Fig. 2 shows.

4 Results

Three different time series have been selected to test our approach that we have called Evolutive Voronoi Predictive system (EVP). The first one represents the water demand in a depot. The second one is an artificial domain widely used in

the literature (Mackey-Glass series), and the last one another one corresponds to natural phenomena (sunspot time series). The first domain has been tested for a prediction horizon of 1 step, in order to try the algorithm for short time predictions. The second domain was selected for two reasons: to test the algorithm in a chaotic time series, and to prove its robustness for long time predictions, as it is tested in bibliography [21, 28]. For the third domain, we preferred to study the performance of the algorithm for different time horizons.

For the prediction process, the algorithm needs to define the minimum number of points per region parameter. This parameter must be set as function of the number of input variables, as it was explained in Sect. 3.4. The standard value for this parameter is 5 per each input variable, so we have done the tests for 3, 5 and 10 per input variable in order to test the robustness of the algorithm. This value will be called regression threshold (3, 5 and 10). The results obtained are displayed in the following subsections.

4.1 Water Depot time series

This time series represents the water demand in a depot, through 5 years and 6 months. The data can be downloaded from http://atc.ugr.es/~jherrera/competicion_sico07.html.

Table 3 Comparative of the error with other algorithms for the Water Depot TS

Algorithm	MSE
EVP	**0.00314**
Regression	0.00379
Conj. Rule	0.02674
IBK	0.00542
Kstar	0.00629
LWL	0.02028
M5Rules	0.00377
M5P	0.00365
Perceptron	0.00673
RBNN	0.00959
SMO Reg	0.00392

The time series has been normalized into the interval [0, 1]. As initial parameter, we decided to use 8 time instants to predict the next one. Then, 1999 patterns were created using the time series. From those, the first 1500 patterns were used to create the training set, and the other 499 for the validation set. The configuration for the experiments is summarized in Table 1.

After this processing task, and with the goal of determining the optimal value of the Voronoi regions, 10 experiments were done for each of the selected values. The mean results for each of these 10 experiments can be seen in Table 2. Each of those experiments where done training 10 subsystems. The column "Reg" is the number of prototypes/regions used. The "Threshold" value is also indicated with the corresponding error and percentage of prediction. The error measure used is the mean squared error. The column "% pred." indicates the percentage of the validation set for which the system produces any prediction. As it can be seen, the percentage of prediction and the error depend both on the minimum number of points required, and also on the number of regions. If the number of regions is high, there are less available points to build each region and therefore, it is more likely that some testing points can not be predicted. However, as Table 2 shows, in most cases a 100% of test patterns are predicted. We can see that the best result is obtained when the threshold value is 10 and the number of regions or prototypes is 3. As we said before, this value corresponds to the mean of ten experiments.

In Table 2 we can see that the error measure obtained is highly independent on the number of points per regions (threshold) and the number of regions. We can also clearly see that the percentage of prediction is very high. In most cases it is 100%, and in the rest of cases is very close to this value. As it could be expected, for high values of the threshold, smaller values of the percentage of prediction are obtained. And that happens when we increase the number of regions. For 20 regions, due to the fact that we are using 1500 training patterns, they cannot be divided into all the regions assuring that each region has, at least, 80 points. Thus, that fact introduces a bias in the results that tends to

Table 4 Statistical test for the Water Depot TS

Regions	Regres	ConjRule	IBK	Kstar	LWL	M5P	M5Rules	Percp	RBNN	SMO-Reg
3	++	++	++	++	++	++	++	+	++	++
6	++	++	++	++	++	++	++	+	++	++
8	++	++	++	++	++	++	++	+	++	++
10	++	++	++	++	++	++	++	+	++	++
12	++	++	++	++	++	++	++	+	++	++
16	++	++	++	++	++	++	++	+	++	++
20	++	++	++	++	++	+	++	+	++	++

decrease the percentage of prediction, but not better prediction results. In summary, experiments proves that the system is able to attain good prediction values that are not very dependant of the algorithm parameters, and a threshold value of 5 is a good parameter for the system.

In Table 3, the best result mentioned above is compared with the results obtained by other well-known machine learning (ML) algorithms implemented in the WEKA tool. For the algorithms that are stochastic, 10 experiments were done, and the table shows the mean of those results. In both tables, the error measure used is the Mean Squared Error (MSE). The EVP row shows the results of the algorithm described in this paper.

A study of the statistical significance of the results was performed by a t-tests, and the results are displayed in Table 4. The alpha level values used were 0.05 and 0.01, and the table compares the results of the EVP algorithm with a threshold value of 10 for different regions with other ML algorithms. The symbols of this table are explained in Table 5. We can see in Table 4 that for a number of regions smaller than 20, EVP is significantly better than most of ML algorithms with an alpha level of 0.01.

Table 5 Values used for Statistical Analysis

Value	Meaning
	Not significant
+	Row algorithm is better than column algorithm with alpha level 0.05
++	Row algorithm is better than column algorithm with alpha level 0.01
− −	Row algorithm is worse than column algorithm with alpha level 0.05
− −	Row algorithm is worse than column algorithm with alpha level 0.01

Table 6 Configuration for the experiments for the Mackey-Glass TS

Domain	Mackey-Glass
Training set	20000
Validation set	1000
Normalization	[0,1]
Input variables	24
Prediction Horizon	50

4.2 Mackey-Glass time series

The Mackey-Glass TS [13, 21, 28] is an artificial series widely used in the domain of the TS forecasting, because it has specially interesting characteristics. It's a chaotic series that needs to be defined with great detail. It is defined by the differential equation (7).

$$\frac{ds(t)}{dt} = -bs(t) + a\frac{s(t-\lambda)}{1+s(t-\lambda)^{10}} \tag{7}$$

As in [13, 21, 28], the values $a = 0.2$, $b = 0.1$ and $\lambda = 17$ were used to generate the TS. 30000 values of the TS are generated for a prediction horizon of 50 using the above equation. The initial 4000 samples are discarded in order to avoid the initialization transients. With the remaining samples, the training set used was composed of the points corresponding to the time interval [5000, 25000]. The test set was composed of the samples [4000, 5000]. All data points are normalized in the interval [0, 1]. This configuration is summarized in Table 6.

The results for the EVP algorithm are shown in Table 7. The best result, corresponding to 40 Voronoi regions and a threshold value of 3 is included in Table 8 and compared with the results obtained by the algorithms IBK, LWL, KStar, M5Rules, M5p, Perceptron Neural Networks, Radial Basis Neural Networks and Conjunctive Rule implemented in WEKA. As in the previous domain, for the stochastic algorithms, 10 experiments were done, and the mean is showed in the table. The error used for the comparison is NRMSE (normalized root mean squared error).

Table 7 shows an uniform behaviour of the system when the threshold is not too high. In that case, the error decreases significantly when the number of regions is increased, but

Table 7 NRMSE and Percentage of Prediction obtained by the EVP for the Mackey-Glass TS

Reg.	Threshold 3		Threshold 5		Threshold 10	
	MSE	% pred.	MSE	% pred.	MSE	% pred.
3	0.2903	100.00	0.2903	100.00	0.2903	100.00
8	0.0730	100.00	0.0730	100.00	0.0735	100.00
12	0.0497	100.00	0.0497	100.00	0.0535	100.00
16	0.0342	100.00	0.0342	100.00	0.0432	98.50
28	0.0253	100.00	0.0260	100.00	0.0459	83.00
32	0.0199	100.00	0.0230	99.40	0.0453	65.30
40	**0.0179**	**100.00**	0.0245	99.30	0.0405	12.90

the percentage of prediction does not change significantly. When the number of regions and the threshold are both increased, the system is not able to find a partition of the input space that keep enough points to fit a regression in each region. As in the previous domain, a threshold between 3 and 5 is enough to attain optimal results, if the number of regions is high enough. As we can see in Table 8, EVP outperforms most of the other algorithms and obtains competitive results when it is compared to K-Star.

Table 9 shows a study of the statistical significance of the results. As in previous domain, the alpha values used were 0.05 and 0.01, and the table compares the results of EVP with other ML algorithms. We can see that for a big number of regions, EVP is significantly better that most ML algorithms, except KStar and IBK.

4.3 Sunspot Time Series

This TS contains the average number of sunspots per month measured from January of 1749 to March of 1977. These data are available at http://sidc.oma.be ("RWC Belgium World Data Center for the Sunspot"). That chaotic TS has local behaviours, noise and even unpredictable zones using the archived knowledge. In Table 11 we can see the error and percentage of prediction obtained by the EVP for different prediction horizons and different population size. Table 12 shows the results obtained for the same horizons by other well-known MLA. SMO-Reg, IBK, LWL, KStar, M5Rules, M5p and Conjunctive Rule are implemented WEKA. The results for Multilayer Feedforward NN and Recurrent NN have been obtained from [7]. The error measure used in both tables is defined in (8)

$$e = \frac{1}{2(N+\tau)} \sum_{i=0}^{N} (x(i) - \tilde{x}(i))^2 \tag{8}$$

In all cases the experiments were done using the same data set: from January of 1749 to December of 1919 for training, and from January of 1929 to March of 1977 for validation, normalized in the [0, 1] interval; in all the cases, 24 inputs were used. The configuration used is shown in Table 10.

In Table 11 we can see that the errors increase with the prediction horizon. However, for each horizon, the error remains in very low margins. If those results are compared with the errors obtained by other algorithms (Table 12), we can see that EVP results are competitive independently of the number of regions, and if we do not consider the percentage of prediction, the results are much better than the ones attained by the other ML algorithms. So we can conclude that EVP has an excellent performance, independent of the prediction horizon.

For the statistical significance analysis (Table 13), horizon 18 and threshold 10 were selected due to their relevance. Alpha values used were 0.05 and 0.01.

Table 8 Comparative of the error with other algorithms for the Mackey-Glass TS

Algorithm	NRMSE
EVP	0.0179
Regression	0.7665
Conj. Rule	0.7202
IBK	0.0208
Kstar	**0.0178**
LWL	0.7004
M5P	0.0503
M5Rules	0.0660
Perceptron	0.1217
RBNN	0.7045
SMO Reg	0.7303

Table 10 Configuration for the experiments for the Sunspot TS

Domain	Sunspot
Training set	2000
Validation set	500
Normalization	[0,1]
Input variables	24
Prediction Horizon	1, 4, 8, 12, 18

Table 9 Statistical test for the Mackey-Glass TS

Regions	Regres	ConjRule	IBK	Kstar	LWL	M5P	M5Rules	Percp	RBNN	SMO-Reg
3	++	++	−−	−−	++	−−	−−	−−	++	++
8	++	++	−−	−−	++	−−	−−	++	++	++
12	++	++	−−	−−	++	++	++	++	++	++
16	++	++	−−	−−	++	++	++	++	++	++
28	++	++	−−	−−	++	++	++	++	++	++
32	++	++	++	−−	++	++	++	++	++	++
40	++	++	++	−−	++	++	++	++	++	++

Table 11 Error measures of the experiments with the EVP for the Sunspot TS

Horizon	Reg.	Threshold 3		Threshold 5		Threshold 10	
		MSE	% pred.	MSE	% pred.	MSE	% pred.
1	3	0.00231	99.86	0.00228	99.80	0.00228	99.64
1	6	0.00224	99.52	0.00221	99.12	0.00220	98.20
1	10	0.00228	98.42	0.00213	96.54	0.00204	92.92
1	16	0.00222	96.48	0.00201	92.80	0.00183	88.28
1	24	0.00197	89.40	0.00180	86.28	**0.00165**	**81.48**
4	3	0.00332	100.00	0.00333	100.00	0.00333	99.98
4	6	0.00334	99.80	0.00335	99.74	0.00330	99.42
4	10	0.00334	99.44	0.00332	99.20	0.00317	96.20
4	16	0.00316	96.50	0.00306	95.74	0.00287	90.98
4	24	0.00289	90.74	0.00266	87.68	**0.00253**	**84.26**
8	3	0.00421	100.00	0.00399	100.00	0.00390	99.60
8	6	0.00408	99.84	0.00396	99.84	0.00391	98.88
8	10	0.00402	99.36	0.00402	98.96	0.00371	96.68
8	16	0.00384	97.86	0.00368	95.26	0.00344	90.92
8	24	0.00377	94.72	0.00354	89.80	**0.00325**	**82.24**
12	3	0.00589	100.00	0.00566	100.00	0.00533	99.30
12	6	0.00548	100.00	0.00540	100.00	0.00523	99.10
12	10	0.00541	99.78	0.00528	99.20	0.00502	96.50
12	16	0.00513	97.74	0.00503	96.10	0.00476	91.56
12	24	0.00509	92.42	0.00478	88.24	**0.00444**	**84.60**
18	3	0.00844	100.00	0.00845	100.00	0.00804	100.00
18	6	0.00817	100.00	0.00807	100.00	0.00777	99.88
18	10	0.00786	100.00	0.00769	99.76	0.00736	96.88
18	16	0.00772	98.84	0.00757	97.96	0.00722	93.12
18	24	0.00738	95.18	0.00722	93.24	**0.00721**	**86.84**

Table 12 Comparative of the error obtained by other MLA for the sunspot TS

Pred. Horiz.	EVP	Linear Regression	Conjunctive Rule	IBK	Kstar	LWL	M5p	M5Rules	SMO-Reg	Feedfw NN	Recurr. NN
1	**0.00165**	0.00230	0.01211	0.00451	0.00503	0.00755	0.00230	0.00230	0.00233	0.00511	0.00511
4	**0.00253**	0.00384	0.01449	0.00564	0.00657	0.01004	0.00484	0.00397	0.00400	0.00965	0.00838
8	**0.00325**	0.00491	0.01594	0.00657	0.00868	0.01233	0.00572	0.00530	0.00520	0.01177	0.00781
12	**0.00444**	0.00636	0.01720	0.00794	0.01038	0.01446	0.00663	0.00642	0.00697	0.01587	0.01080
18	**0.00721**	0.00989	0.02040	0.01039	0.01312	0.01894	0.00807	0.00742	0.01131	0.02570	0.01464

Table 13 Statistical test for the Sunspot TS

Regions	Regres	ConjRule	IBK	Kstar	LWL	M5P	M5Rules	SMO-Reg
3	++	++	++	++	++		−−	++
6	++	++	++	++	++	++	−−	++
10	++	++	++	++	++	++		++
16	++	++	++	++	++	++	++	++
24	++	++	++	++	++	++	++	++

5 Summary and conclusions

There are many methods able to tackle time series prediction problems, but in most cases these methods only look for a general approach for the series behavior. Time series generally have local behaviors that do not allow to obtain good predictions using a global approach. Our work focuses on finding these local behaviors in the time series, allowing to achieve more accurate predictions.

The method is based on the division of the input space into Voronoi regions using Evolution Strategies. In each of these regions, a linear regression is performed. In order to obtain more reliable predictions, a minimum number of data points is required in each Voronoi region. Thus, some regions might not reach the minimum number of points required being unable to make the prediction for the test pattern. With the aim of avoiding this situation and to increase the prediction accuracy, instead of building a model for the whole input space, several models or subsystems are built with different random initializations. In this way, the prediction for a given testing pattern will be the mean of the valid predictions of the different subsystems. It might happen that, for a given test point, some model can not make a prediction but others will make it. So, the whole system, composed by several prediction subsystems, will be able to produce an output.

We have applied our method to a different domains: a Time Series representing the monthly demand of water in a specific depot through a long period of time, a well-known chaotic time series that has been widely used in the literature (Mackey-Glass TS), and another one modelling a real phenomenon (Sun Spot TS). We have tested the method performance with different prediction horizons. Thus, we have done the experiments with a small horizon (horizont 1) for the Water Depot TS, a big horizon value (50) for the Mackey-Glass TS and several horizon values for the Sun spot TS. Besides, in order to analyze the influence of the method parameters, we have used different number of Voronoi regions and different regression threshold values (3, 5 and 7) corresponding to a minimum amount of points required to carry out the regression on each region.

For comparative purposes, ten different and commonly used machine learning algorithms have been applied to each problem. The results obtained show that in all the domains the proposed method obtains, in most situations, better results than the other algorithms, for all the prediction horizons tested, being able to predict a 100% of the testing points in most cases. It can be observed that the number of regions and the regression threshold are not critical parameters because the method obtains quite steady results in all situations. Of course, when the number of training patterns is small and the number of regions and the regression threshold increase, then more regions are unable to build the regression model and more testing points cannot been predicted. Thus the percentage of prediction decreases.

Our algorithm has also some additional advantages. Although it has been designed to solve time series problems, it can be applied to any prediction problem that can be represented by patterns. Another advantage is that it can use any predictive system associated to each region, such as multilayer perceptron networks, radial basis NN, etc. rather than linear regression, depending on the characteristics of the problem.

References

1. Bäck T, Hoffmeister F, Schwefel H-P (1991) A survey of evolution strategies. In: *ICGA*, pp 2–9
2. Bäck T, Schwefel H-P (1992) Evolutionary algorithms: Some very old strategies for optimization and adaptation. In: Proc. second int'l workshop software engineering, artificial intelligence, and expert systems for high energy and nuclear physics, pp 247–254
3. Booker LB, Goldberg DE, Holland JH (1989) Classifier systems and genetic algorithms. Artif Intell 40(1–3):235–282
4. Box GEP, Jenkins GM, Reinsel GC (1976) Time series analysis: forecasting and control. Holden-Day, Oakland
5. Fernández F, Isasi P (2008) Local feature weighting in nearest prototype classification. IEEE Trans Neural Netw 19(1):40–53
6. Fogel DB (1994) An introduction to simulated evolutionary optimization. IEEE Trans Neural Netw 5(1):3–14
7. Galván IM, Isasi P (2001) Multi-step learning rule for recurrent neural models: An application to time series forecasting. Neural Process Lett 13(2):115–133
8. Galván IM, Isasi P, Aler R, Valls JM (2001) A selective learning method to improve the generalization of multilayer feedforward neural networks. Int J Neural Syst 11(2):167–177
9. Holland JH (1975) Adaptation in natural and artificial systems. MIT Press, Cambridge
10. Lloyd SP (1982) Least square quantization in pcm. IEEE Trans Inf Theory 2(28):129–137
11. Luque C, Isasi P, Hernández J (2004) Distribución de cargas en una esfera mediante estrategias evolutivas. Rev IEEE Am Lat 2(2)
12. Luque C, Isasi P, Hernández JC (2004) Forecasting time series by means of evolutionary algorithms. In: *PPSN*, pp 1061–1070
13. Mackey M, Glass L (1977) Oscillation and chaos in physiological control systems. Science 197:287–289
14. Macqueen J (1967) Some methods for classification and analysis of multivariate observations. In: 5th Berkeley symposium on mathematical statistics and probability, Berkeley, January 1967
15. Meyer TP, Packard NH (1992) Local forecasting of high dimensional chaotic dynamics
16. Mitchell M (1996) An introduction to genetic algorithms. MIT Press, Cambridge
17. Moody J, Darken C (1989) Fast learning in networks of locally tuned processing units. Neural Comput 1:281–294
18. Packard NH (1990) A genetic learning algorithm for the analysis of complex data. Complex Syst 4(5):543–572
19. Papalexopoulos A, Hesterberg T (1990) A regression-based approach to short-term system load forecasting. IEEE Trans Power Syst 5(4):1535–1547
20. Park D, El-Sharkawi M, Marks I, Atlas L, Damborg M (1991) Electric load forecasting using an artificial neural network. IEEE Trans Power Syst 6(2):442–449

21. Platt J (1991) A resource-allocating network for function interpolation. Neural Comput 3:213–225
22. Quintana D, Luque C, Isasi P (2005) Evolutionary rule-based system for IPO underpricing prediction. In: Proceedings of genetic and evolutionary computation conference (GECCO 2005), pp 983–989
23. Rahman S, Hazim O (1993) A generalized knowledge-based short-term load-forecasting technique. IEEE Trans Power Syst 8(2):508–514
24. Schwefel HP (1965) Kybernetische Evolution als Strategie der exprimentellen Forschung in er Strömungstechnik. PhD thesis
25. Somervuo P, Kohonen T (1999) Self-organizing maps and learning vector quantization for feature sequences. Neural Process Lett 10(2):151–159
26. Valls JM, Galván IM, Isasi P (2008) Learning radial basis neural networks in a lazy way: a comparative study. Neurocomputing 71:2529–2537
27. Wei H, Billings S (2006) An efficient nonlinear cardinal b-spline model for high tide forecasts at the Venice lagoon
28. Yingwei L, Sundararajan N, Saratchandran P (1997) A sequential learning scheme for function approximation using minimal radial basis function neural networks. Neural Comput 9:461–478
29. Zaldivar J, Gutiérrez E, Galván I, Strozzi F, Tomasin A (2000) Forecasting high waters at Venice Lagoon using chaotic time series analysis and nonlinear neural networks. J Hydroinf 2:61–84
30. Zuo J, Tang C, Li C, an Yuan C, long Chen A (2004) Time series prediction based on gene expression programming. In: Li Q, Wang G, Feng L (eds.) Advances in web-age information management: 5th international conference, WAIM 2004, Lecture notes in computer science, vol 3129, pp 55–64, Dalian, China, 15–17 July 2004. Springer, Berlin